EXPERIÊNCIA 8:

LINGUAGEM DE ALTO NÍVEL "C" PARA 8051 E GRAVAÇÃO DE PROGRAMAS EM EPROM

Autores: Prof. Dr. André Riyuiti Hirakawa e Prof. Dr. Carlos Eduardo Cugnasca

Versão 3.0 - 2009

1. OBJETIVO

Esta experiência tem como objetivo complementar o aprendizado de elaboração de programas em linguagem de alto nível, apresentando os aspectos do acesso a periféricos, a sinais de controle e a interrupções. Além da programação do microprocessador em linguagem de alto nível, essa experiência prevê a gravação do programa executável em memória EPROM, que deverá substituir a memória com o Programa Monitor da Placa Experimental.

2. INTRODUÇÃO

Os compiladores são ferramentas poderosas para a produção de software, por traduzirem programas escritos em linguagem de alto nível para programas escritos em linguagem de máquina.

Entretanto, a interação da linguagem de alto nível com as operações sobre o hardware (operações de entrada e saída), em alguns casos, não são previstas na linguagem. Nos programas executados em computadores, as operações de entrada e saída são executadas pelo sistema operacional, através de "*systems calls*". Nos equipamentos desprovidos de sistema operacional é necessário que o programador crie os programas básicos de entrada e saída, escritos usualmente em linguagem assembly para que apresentem um bom grau de otimização, minimizando o tempo de sua execução. Eles serão utilizados por outros programas, escritos em linguagem de alto nível ou mesmo em linguagem assembly, e são comumente agrupados em bibliotecas.

Esses programas são escritos na forma de *subrotinas*, e é altamente conveniente que sejam respeitados os **padrões de passagem de parâmetros e devolução de resultados** definidos pelo compilador que se pretende utilizar, visando as suas chamadas dentro de programas escritos em linguagem de alto nível.

Esse conjunto de subrotinas básicas deve ser anexado ao programa que as utilize na fase de **ligação/alocação**. Cada subrotina deve, portanto, ser declarada como **pública** para que sejam satisfeitas as chamadas geradas pelo programa em linguagem de alto nível, que a declarará como **externa**. De forma análoga, pode-se desenvolver rotinas em C e chamá-las em programas elaborados em linguagem assembly.

3. CARACTERÍSTICAS DO COMPILADOR C

Cada compilador costuma apresentar características próprias para a geração das instruções em linguagem de máquina, tais como passagem de parâmetros para subrotinas, devolução de resultados de funções, tratamento de interrupções, etc. Assim, **compiladores de fabricantes diferentes, que geram código para um mesmo processador, não necessariamente adotam as mesmas convenções**.

Também muitos recursos existentes em uma implementação convencional de C utilizada em microcomputadores, nem sempre são encontrados em implementações especificas para a geração de código para determinados microprocessadores (por exemplo, 8080/8085, Z80, 8086/8088, 68000, etc).

Com o advento do microcomputador pessoal surgiram compiladores C específicos para essas máquinas, com bibliotecas de rotinas de manipulação dos seus periféricos típicos (teclado, vídeo, disco, etc), e gerando o programa executável compatível com elas, e que nem sempre permitem que o mesmo possa ser utilizado para a geração de programas para sistemas diferentes, baseados no mesmo processador (por exemplo, baseados no 8086/8088). Tais sistemas costumam apresentar áreas de memória especificas, diferente dos computadores pessoais, exigindo compiladores e outras ferramentas de geração de programa, com capacidade de alocação de programas para qualquer região de memória escolhida pelo programador.

Os manuais específicos de cada compilador devem sempre ser consultados, observando-se as restrições existentes, os recursos de biblioteca disponíveis, os mecanismos de geração de código e demais convenções adotadas.

O Apêndice I apresenta as principais características do compilador C a ser utilizado [1]. Informações genéricas a respeito da linguagem C podem ser obtidas em [4], [5], [6], e [7].

4. PARTE EXPERIMENTAL

4.1 Planejamento

O planejamento deverá apresentar:

- Descrição do projeto, relacionando suas características principais.
- O estudo do manuseio de interrupções em linguagem “C” apresentado em [1].
- Utilização do Diagrama Estruturado de Nassi-Schneirderman (ou Carta de Nassi-Schneirderman [10]).
- Especificação do programa a ser testado, com diagramas estruturados e linhas de programas com explicação sobre qual a função da execução de cada comando. **Planejamento sem esta documentação não será aceito!**
- Edição do arquivo-fonte: existem editores que facilitam a edição de programas bem como a impressão da listagem associada, como por exemplo, o Crimson Editor [11].

4.2 Atividades

a) Refazer o programa do relógio da **Experiência 5** em linguagem "C". Utilize as rotinas auxiliares da experiência anterior para depuração do programa (Veja as informações do Apêndice I para a utilização da interrupção em Linguagem "C", especialmente a partir do item j.).

   OBS: Como os programas em linguagem "C" costumam criar programas executáveis de tamanhos significativamente maiores do que se fossem desenvolvidos em linguagem assembly, o tempo disponível para o tratamento de interrupção se torna mais crítico. Assim, deve-se evitar a colocação de muitas instruções dentro de subrotinas de tratamento de interrupções.

b) Gerar uma versão executável do programa anterior para ser gravada em EPROM. Procure seguir as recomendações abaixo:

   - O endereço inicial do programa gerado deve ser **0000H**.
   - Uma das tarefas do Programa Monitor é inicializar os registradores e os periféricos do 80C51. Com a retirada desse programa, isso não mais será feito. Portanto, o programa do relógio nesta versão, deve inicializar adequadamente alguns dos registradores de configuração do microcontrolador (por exemplo, **PSW, IE, IP** e **SP**). Descubra no manual do compilador como essa questão pode ser resolvida, e coloque no relatório.
   - Deve-se tomar cuidado com os parâmetros usados no processo de compilação e ligação/alocação (por exemplo, parâmetros de otimização e modelo – *small, large* – veja Apêndice I).
   - **Interrupções:** o Programa Monitor possui instruções de desvio (**LJMP**) para o final da memória RAM externa para cada posição do seu vetor de interrupções, redefinindo-o assim. Assim, permite-se que sejam colocadas subrotinas de tratamento de interrupções em qualquer local, sem a necessidade de se alterar o conteúdo da memória EPROM. A tabela abaixo apresenta o novo vetor disponível. Contudo, neste item, tal característica não será utilizada.

Interrupção		Endereço original	Endereço na Placa Experimental
IE0	Interrupção externa 0	0003H	FFF0H
TF0	Timer/Contador 0	000BH	FFF3H
IE1	Interrupção externa 1	0013H	FFF6H
TF1	Timer/Contador 1	001BH	FFF9H
RI + TI	Canal Serial	0023H	FFFCH

c) Efetue uma pesquisa em manuais e *sites* de fabricantes de memória e/ou livros especializados, e **coloque no relatório** uma pesquisa a respeito do funcionamento de uma memória EPROM, e seu processo de gravação e de apagamento (**não esquecer de citar as fontes de consulta**). Responda ainda às seguintes questões:

   - Qual é o tempo típico de gravação e de apagamento de uma memória?
   - O que é uma memória do tipo OTP?
   - Quais são as diferenças entre as memórias do tipo ROM, PROM, EPROM, EEPROM, E^2PROM e FLASH?

d) Refazer o programa experiência **"IMPLEMENTAÇÃO DE RELÓGIO DIGITAL COM INTERRUPÇÃO EM LINGUAGEM "C".** O cronômetro deve possuir pelo menos uma tecla para disparar e parar a contagem. A implementação das teclas de **RESET** e **LAP**, normalmente encontradas nos relógios digitais é opcional. Utilize as subrotinas de depuração e apresentação

no display desenvolvidos na experiência anterior e o módulo de identificação de teclado reescrito em linguagem “C”.

e) Envie a informação apresentada no display também no terminal de vídeo.

f) Gerar uma versão executável do programa anterior para ser gravada em EPROM, e testá-la.

g) **Opcionais** (não é necessário gravar novamente a EPROM):

- Altere o programa para que o teclado de terminal de vídeo substitua o teclado da Placa Experimental.
- Utilize o código do sinal sonoro do terminal de vídeo (BEL) para indicar a parada e o reinício, com *bips* de duração diferente.

5. BIBLIOGRAFIA

[1] SDCC - Small Device C Compiler, http://sdcc.sourceforge.net/ .

[2] Andrade, Marco Túlio Carvalho de; Cugnasca, Carlos Eduardo; Hirakawa, André; Apostila da Experiência LINGUAGEM DE ALTO NÍVEL “C” PARA 8051 E GRAVAÇÃO de programas EM EPROM, Disciplina PCS 597 - Laboratório de Microprocessadores I, 2001.

[3] Cugnasca, Carlos Eduardo; Zerbini, ricardo Costa; Apostila da Experiência Desenvolvimento de Programas em Linguagem de Alto Nível “C”, Disciplina PCS 599 - Laboratório de Microprocessadores, 1.995.

[4] LEVENTHAL, L. A. 8080A-8085 ASSEMBLY LANGUAGE PROGRAMMING. McGraw-Hill. 1986.

[5] KERNIGHAN,B.W. & RITCHIE,D.M.C. A Linguagem de Programação. Padrão ANSI. Rio de Janeiro, Editora Campus, 1990. 289p.

[6] CUGNASCA, C. E. Linguagem PL/M. Apostila FDTE, São Paulo, 1986.

[7] PRESSMAN R. S. Engenharia de Software, Makron Books, 1995.

[8] PHILIPS; 80C51-Based 8-Bit Microcontrollers – Data Handbook IC20, Philips Electronics North America Corporation, USA, 1.997.

[9] ALFACOM; Módulos Multi-Matrix – Manual de Utilização, Editora Érica Ltda., 1.994.

[10] EasyCode - http://www.easycode.de/ ou http://www.blichfeldt.dk/ipabxwpp/.

[11] Crimson Editor - http://www.crimsoneditor.com/

APÊNDICE I – ALGUMAS CARACTERÍSTICAS DO SMALL DEVICE C COMPILER

Recomenda-se a leitura atenta do manual desse compilador. A seguir, serão apresentadas algumas informações de interesse ao programador. Maiores informações, bem como o manual do compilador podem ser obtidos em http://sdcc.sourceforge.net/.

Em relação aos exemplos apresentados na apostila há a necessidade de se realizar algumas alterações, sugerindo-se fortemente a leitura atenta do manual disponível no endereço acima.

A seguir são apresentadas algumas recomendações para seu uso.

a) Declarações.

Para especificar áreas de memória utilizar `at`. Exemplo: definição da variável `var` na posição `98H` da área **SRF** do 80C51,

```
unsigned char near at 0x98 var;
```

b) Uso de Linguagem Assembly dentro da Linguagem C.

Embora possível, **não é recomendável**, pelos efeitos colaterais que pode causar (por exemplo, alteração de valores de registradores que o compilador também utiliza).

```
_asm
 <código em linguagem assembly>
_endasm;
```

Observação: o `;` da última instrução não pode ser esquecido, bem como o símbolo `_`.

c) Chamada do Compilador.

Utilizar o comando SDCC em uma janela DOS, colocando-se as informações de alocação na mesma linha. Exemplo: compilação do programa `progr.c`, alocando a área de código em `A000H` e a área de dados em `8000H`.

```
c:>sdcc --code-loc 0xA000 --xram-loc 0x8000 progr.c
```

d) Passagem de Parâmetros e Retorno de Valores em Subrotinas.

Alguns registradores são utilizados tanto para a passagem de parâmetros como para o retorno de valores. Um byte: `DPL`; dois bytes; `DPL`, `DPH`; três bytes: `DPL`, `DPH`, `B` e quatro bytes: `DPL`, `DPH`, `B` e `A`.

Para os parâmetros seguintes, caso a subrotina seja não **reentrante**, os parâmetros são passados pela memória; em caso de subrotinas **reeentrantes**, pela pilha (para maiores detalhes, consultar documentação do SDCC, item 3.14.1).

e) Declaração de uma Função Escrita em Linguagem Assembly Presente em um Arquivo Externo.

Exemplo: rotina no arquivo `rot.asm`:

extern tipo_retorno nome_função (tipo_parametro1, tipo_parametro2, ...);

Os nomes dos parâmetros não devem ser passados.

Na rotina deve ser declarado que a rotina é de uso global e que o código é relocável. Exemplo:

```
.globl _delay

.area CSEG

_delay: ...

        RET
```

Observação: não se utiliza a pseudo-instrução `END`.

No caso de a subrotina possuir, por exemplo, um parâmetro, o mesmo deve ser declarado. Exemplo:

```
.globl _delay_PARAM_1

.globl _delay

.area CSEG

_delay: ...
```

RET

Observação: não se utiliza a pseudo-instrução `END`.

Deve-se primeiramente chamar o montador para gerar o código da função, e depois o compilador para gerar o código do programa que invoca a função e ligá-lo com o código deste. Exemplo:

```
c:>asx8051 -losg rot.asm       → gera o arquivo rot.rel

c:>sdcc --code-loc 0xA000 --xram-loc 0x8000 progr.c rot.rel
```

- a opção `--code-loc 0xA000` aloca o código a partir do endereço `A000H.`
- a opção `--xram-loc 0x8000` aloca a área de variáveis externas a partir do endereço `8000H.`
- `além do rot.rel` outros arquivos relocáveis escritos em linguagem assembly/C que já foram montados/compilados poderiam ser acrescentados (inclusive bibliotecas prontas ou criadas pelo usuário); ele será ligado ao programa `progr.c` que está sendo compilado através deste comando, e o resultado será alocado, gerando o programa absoluto `progr.ihx` no formato hexadecimal Intel (a ser transferido para aplaca experimental);

outros arquivos .rel poderiam ser colocados neste comando, sendo que o primeiro da lista (no caso `progr.c` deve necessariamente conter o `main`.

f) Includes.

O SDCC permite que includes sejam utilizados no início de cada programa visando simplificar a sua escrita, sendo que um deles é sempre utilizado, pois contém as declarações dos registradores, portas e demais recursos do 8051, que é o arquivo `8051.h`. Exemplos:

```
#include <MCS51/8051.h>
```

Nesse caso o arquivo tipo texto `8051.h` pode ser localizado por meio do caminho `SDCC/MCS51/8051.h`. Para o caso de o arquivo se encontrar no mesmo diretório do programa que está sendo compilado, tem-se a opção:

```
#include "8051.h"
```

g) Tipos de Variáveis e Classes de Armazenamento

O SDCC permite diversos tipos de variáveis, como `bit`, `char` (8bits, 1 byte), `short e int` (16 bits, 2 bytes), `long` (32 bits, 4 bytes) e `float` (4 bytes, padrão IEEE). As variáveis do tipo `bit` são armazenadas na memória RAM interna (endereços de 0x20 a 0x2f). Já as demais variáveis podem ser armazenadas na RAM interna (`data` ou `near`), na RAM externa (`xdata` ou `far`), ou na área de código (`code`). O SDCC especifica outras classes de armazenamento além das especificadas pelo Padrão ANSI, para adequar a linguagem às particularidades de cada microcontrolador que ele suporta. Para o 8051, tem-se (item 3.4.1 de [1]):

```
data/near        code
xdata/far        bit
idata            sfr/sbit
pdata            ponteiros
```

h) Ponteiros.

O SDCC permite alguns tipos de ponteiros, como por exemplo:

- ponteiro armazenado na memória RAM interna apontando para dados na memória RAM externa:

  ```
  xdata unsigned char *data p;
  ```

- ponteiro armazenado na memória RAM externa apontando para dados na memória RAM interna:

  ```
  data unsigned char *xdata p;
  ```

Para maiores detalhes consultar item 3.4.1.8 do manual do SDCC.

i) Modelos de Compilação.

Uma vez que o microcontrolador 8051 permite o uso de memórias ROM e RAM internas ou externas, foram previstos dos modelos se compilação, que permitem aos projetistas alocar as instruções e área de dados de forma compatível com o previsto no projeto. Maiores informações, capítulo 3.17 de [1]. No comando de compilação pode ser acrescentada uma das opções:

- `--model-large`
- `--model-small (default)`

j) Subrotinas de Interrupção.

Uma subrotina de tratamento de interrupção deve ser declarada de forma semelhante à da subrotina abaixo, específica para o Timer 0 do 8051:

```
void timer0_int (void) interrupt 1 using 1
{
. . .
}
```

O número da interrupção é atribuído da seguinte forma:

N. da interrupção	Descrição	Endereço do vetor	Endereço de desvio na Placa Experimental
0	Interrupção Externa 0	0x0003	0xFFF0
1	Timer 0	0x000B	0xFFF3
2	Interrupção Externa 1	0x0013	0xFFF6
3	Timer 1	0x001B	0xFFF9
4	Canal Serial	0x0023	0xFFFC
5	Timer 2 (8052)	0x002B	-

O comando `using` especifica o banco de registradores a ser utilizado na subrotina de tratamento de interrupção. Maiores informações, capítulo 3.8 de [1].

k) Bibliotecas.

O SDCC dispõe de bibliotecas prontas para uso (capítulo 3.16 de [1]). Também permite que o programador crie as suas próprias bibliotecas (capítulos 3.13 e 3.14 de [1]).

l) Outros Recursos.

O capítulo 8 de [1] apresenta uma série de comandos úteis ao programador, como por exemplo, deslocamento e rotação de bits, otimizações, etc.

APÊNDICE II - GRAVAÇÃO DE EPROM 27256 VIA PC UTILIZANDO O MPT-2000

Passos a serem seguidos para a gravação de memórias do tipo EPROM:

1. Ligar o equipamento de gravação de dispositivos **MPT-2000** (ele executará automaticamente um auto teste).
2. Selecionar tipo do dispositivo (neste caso uma EPROM): para selecionar uma função qualquer basta pressionar a tecla **TEST.**
3. Mover o cursor até a opção "**0**" (8 bit EPROM) e pressionar **TEST**.
4. Examinar a memória a gravar, identificando o seu fabricante, e seu tipo (por exemplo, Intel, 27C256).
5. Selecionar o fabricante da memória.
6. Selecionar o número do dispositivo (neste caso 27256 ou 27C256)
7. Pressione **CHECK** para verificar se o componente está realmente apagado (uma memória EPROM é considerada apagada quando todas as suas posições apresentam o valor **FFh**). Caso ela não esteja apagada, coloque-a no Apagador de Memórias por alguns minutos.

Operação no PC:

Realizar o boot do PC no modo DOS (tecla F8 pressionada durante o boot).

8. Inserir disco MPT-2000 (ou selecionar o diretório do disco rígido do microcomputador que possua os programas do equipamento MPT-2000) e digitar **MAIN**.
   C: CD\LAMI\MINIPA\MAIN

9. Digitar **ENTER** duas vezes.
10. Digitar [**l**] para carregar arquivo do disco.
11. Selecionar o formato do arquivo (por exemplo: **INTEL HEX**).
12. Especificar nome do arquivo a ser carregado (por exemplo: **c:\teste\exp5.hex**).
13. **ENTER**.
14. Digitar "**F**".
15. **ENTER**.
16. **ENTER** (selecionar opção **PC => SU1**).
17. **ENTER (MEMORY BUFFER**).
18. **ENTER (START)**.

Obs. : executar os passos 20, 21 e 22 (preparar terminal) antes de finalizar.

19. **ENTER (END)**.

Operação do terminal MPT-2000:

20. Pressionar a tecla **FUNC**.
21. Selecionar opção "**B**" (up/down-load data).
22. Selecionar opção "**1**" (download **PC => SU1**).

Os passos executados anteriormente armazenam o programa no buffer do MPT-2000. Para que este seja transferido para a EPROM basta digitar a tecla **PROG** do equipamento de gravação.